REMARKS

This application has been reviewed in light of the Office Action dated September 20, 2006, made Final by the Examiner. Claims 1-30 are pending in the application. By the present amendment, claims 1 and 22 have been amended. No new matter has been added. The Examiner's reconsideration of the rejection in view of the amendment and the following remarks is respectfully requested.

By the Office Action, claims 1-30 stand rejected under 35 U.S.C. §102 (b) as being anticipated by U.S. Patent No. 5,765, 035 to Tran (hereinafter Tran).

Tran is directed to a system having a reorder buffer that detects dependencies between accesses to caches. Tran does not disclose or suggest the stages or operations as set forth in the present claims. While Tran deals with a common subject matter, i.e., processor pipelines, Tran does not include all of the elements as set forth in the claims. Tran discloses indirect addressing but fails to disclose or suggest pointer checking in advance of dependency checking. Further, Tran does not disclose or suggest pointer analysis or pointer register files, which include pointers and updates as set forth in the present claims. Although claim 1, as previously presented, is believed to have sufficient merit to permit an allowance, claim 1 has been amended to further distinguish it over Tran and provide clarity.

Tran contemplates a dependency checking structure for detecting dependencies between accesses to a pair of caches (See Summary of Tran). The dependency checking structure comprises a reorder buffer 216, which is located after the issue stage 208. (Dependency checking may also be performed in the Load/Store unit 222, which is also after the issue stage 208.) The reorder buffer 216 includes a request bus, a plurality of storage locations, a plurality of comparator circuits, and a control unit. The request bus is configured to

convey an access address. Each of the storage locations is configured to store information pertaining to an instruction, wherein the information may include an address of an operand. Coupled to a respective storage location, each of the comparator circuits is configured to receive the address stored therein.

Additionally, each of the comparator circuits is coupled to the request bus. The comparator circuits are configured to produce a comparison signal indicative of equality between a field of the address stored in the associated storage location and a corresponding field of the access address from the request bus. Coupled to the storage locations, the request bus, and the comparator circuits is the control unit. The control unit receives the comparison signals from the comparator circuits and is configured to convey a dependency signal indicative of a dependency between the access address and the address stored in the respective storage location. The control unit conveys the dependency signals upon a dependency bus. If the execution stage's read memory access is dependent on a write memory access performed by the decode stage, the read access is stalled until the write access completes. If the read depends on the write access, the memory is flushed.

While the system of Tran provides for stalling an instruction before all information needed for the instruction is available, Tran does not disclose or suggest pointer dependency checking in advance of instruction issuance, or other elements of the present claims as will be described below.

Claim 1 includes, *inter alia*, a pointer register stage which stores pointer information and updates, a pointer dependency checking stage located downstream of the pointer register stage, which determines if instruction pointer dependencies exist and stalls an instruction prior to issuance if necessary to resolve inter-instruction dependencies, at least one

functional unit providing pointer information updates to the pointer register stage such that the pointer information is processed and updated to the pointer register stage before an instruction goes through the dependency checking stage, an issue stage, and an execution stage.

In particular, Tran includes an issue unit 208 before any of its register file 218, reorder buffer 216, reservation station units 210 and functional units 212. A pointer register file which provides pointer listings and updates is not disclosed in Tran. In accordance with the present invention, the pointer register stage includes pointer information that is updated by the functional unit. There are no pointer register stages in Tran. The Examiner states that col. 9, lines 32-37 of Tran teaches a pointer register stage. The Applicant disagrees.

The reservation station units 210A-210F temporarily store "bit-encoded instructions" waiting to be executed by one of the functional units and values of operands for these instructions (see col. 9, lines 32-37). These instructions have already been issued and are waiting operands.

The Examiner further contends that indirect addressing may be used. Col. 7 lines 31-57 merely discloses what indirect addressing is and that indirect addressing for operands may be employed. However, this does not make the reservation station units 210A-210F, pointer register files. The issued instructions are not pointers even if the operands are located using indirect addressing. In particular, this configuration would suffer from the problems trying to be solved by the present invention. Namely, the pointer register file is not updated in advance of the issuance of these instructions, because a pointer analysis has not been done in advance of the issuance. The reservation station units 210A-210F are therefore not pointer register files.

According to the present invention, one deficiency of the prior art implementations of an indirect register access mechanism is that no checking is performed for register dependencies between instructions (that is, dependencies through registers in a register file). These dependencies cannot be checked before the pointers are accessed, because the values of indexes to the register file are not known at the dependence checking stage, and different pointers may point to the same entry in the register file (i.e., aliasing problem).

The present claims make the pointer updates to the pointer register file available to the instruction an subsequent instructions with low latency (the latency could be as low as a single cycle), resulting in a reduced frequency of bubbles (or reduced number of unused issue slots) in instruction sequences with frequent inter-instruction dependencies through register pointers. By running through the pointer information in advance of instruction issuance, delays in the pipeline can be reduced. Tran does not address this problem or provide the structure to address this problem. Therefore, a pointer register stage is not disclosed or suggested by Tran.

Claim 1 now recites: a pointer dependency checking stage located downstream of the pointer register stage, which determines if instruction pointer dependencies exist and stalls an instruction prior to issuance if necessary to resolve inter-instruction dependencies. Even if register file 218 is considered, *arguendo*, a pointer register stage, the instructions have already been issued by issue unit 208, and so pointer dependencies may not be used to stall instructions before issuance.

Claim 1, further recites: at least one functional unit providing pointer information updates to the pointer register stage such that the pointer information is processed and updated to the pointer register stage before an instruction goes through the dependency checking stage, an issue stage, and an execution stage. Tran clearly does not process and

update pointer information and provide the pointer update information to the pointer register stage before an instruction goes through the dependency checking stage, an issue stage, and an execution stage. No disclosure or suggestion of this element is provided in Tran. Tran provides a reorder buffer to check dependencies of instructions that have already been issued by issue unit 208. The only thing left to do in the reservation stations 210 is to wait for operands and execute the instructions. In Tran, pointer dependencies are not checked in advance of issuance.

Although the Applicant believes that claim 1 is allowable over the cited art, claim 1 has been amended to provide further clarity. Reconsideration of the rejection is earnestly solicited.

Regarding claim 13, claim 13 recites, *inter alia*, a pipeline ...including ... a pointer register stage which stores pointer information and updates; a dependence checking stage located downstream of the pointer register stage, which determines if instruction dependencies exist and stalls an issue prior to issuance if necessary to resolve inter-instruction dependencies; a pointer execution stage for processing pointers prior to the dependence checking stage, the pointer execution stage providing pointer updates to the pointer register stage via an early pointer update path; and at least one functional unit providing pointer information updates to the pointer register stage such that pointer information is processed and updated to the pointer register stage.

In addition to the arguments set forth for claim 1, claim 13 further provides a pointer execution stage for processing pointers prior to the dependence checking stage and an early pointer update path. These elements are not disclosed or suggested by Tran. The Examiner states that the function units (reservation units) are part of a pointer execution stage.

This is not the case. The Examiner cites col. 9 lines 45-50, which sets forth "results forwarding" for the actual instruction. In Tran, pointers and pointer updates are not sent to a pointer register file before the issuance of commands. Instead, instruction results are forwarded to already-issued instructions that are waiting for a result of instruction execution. In Tran, there is no pointer analysis performed in advance of issuing an instruction or an early pointer update path disclosed or suggested, as set forth in claim 13. Reconsideration of the rejection is earnestly solicited.

Regarding claim 22, claim 22 recites, *inter alia*, a method for updating pointers ahead of an instruction, includes providing a plurality of operational stages, including a pointer register stage ...., a pointer dependence checking stage located downstream of the pointer register stage, which determines if instruction dependencies exist and stalls an issue prior to issuance if necessary to resolve inter-instruction dependencies, and at least one functional unit providing pointer information updates to the pointer register stage and processing pointer information to update the pointer information for the pointer register stage so that updated pointer information is available such that the pointer information is processed and updated to the pointer register stage before an instruction goes through the dependency checking stage, an issue stage, and an execution stage.

As provided above, the pointer dependencies are checked in advance of issuance on an instruction. This is equivalent to issuing an instruction and waiting for operands as described in Tran. The previous arguments are applicable to claim 22, which has been amended for clarity.

The Applicant notes with appreciation, the *Response to Arguments* section, where the interpretation of "issuance" is described. While the term "issuance" may mean issuance to a particular stage, it is believed the "prior to issuance" as set forth herein means before any issuance. This is supported by the FIGS. and accompanying text (see e.g., FIG. 2), and is believed to be further clarified by the present amendments. Since pointer dependencies are checked in advance of issuance, the instruction is assured the most up to date pointer information.

Therefore it is respectfully submitted that Tran fails to disclose or suggest all of the elements of claims 1, 13 and 22 as amended. Claims 1, 13 and 22 are believed to be in condition for allowance for at least the stated reasons. Dependent claims 2-12, 14-21 and 23-30 are also believed to be allowable for at least their dependencies from claims 1, 13 and 22, respectively. Reconsideration of the rejection is earnestly solicited.

Other reasons exist for allowing the dependent claim as well. For example, dependent claim 2 recites, *inter alia*, a pointer execution stage used before the dependency checking stage such that inter-instruction dependency is checked after the pointer execution stage or in parallel with pointer execution. Tran does not perform pointer execution in addition there is no pointer execution stage used before the dependency checking stage. Similarly, the elements of claim 5, which recites a pointer execution bypass making pointer updates available to immediately following instructions before a pointer update is written into the pointer register file are not disclosed or suggested.

The Applicant believes that a telephone conference to discuss that case would be beneficial, and invites that Examiner to contact the undersigned.

In view of the foregoing amendments and remarks, it is respectfully submitted that all the claims now pending in the application are in condition for allowance. Early and favorable reconsideration of the case is respectfully requested.

It is believed that no additional fees or charges are currently due. However, in the event that any additional fees or charges are required at this time in connection with the application, they may be charged to applicant's IBM Deposit Account No. 50-0510.

Respectfully submitted,

Date: 11/17/06

By: _____
James J. Bitetto
Registration No. 40,513

Mailing Address:

KEUSEY, TUTUNJIAN & BITETTO, P.C.
20 Crossways Park North, Suite 210
Woodbury, NY 11797
Tel: (516) 496-3868
Fax: (516) 496-3869